

**AMC4030运动控制器**

**编**

**程**

**手**

**册**

**（1.0版）**

**成都福誉科技有限公司**

**FUYU TECHNOLOGY CO.,LTD.**

**版权申明**

**成都福誉科技有限公司**

**保留所有权利**

**成都福誉科技有限公司保留在不事先通知的情况下，修改本手册中的成品和产品规格等文件的权利。**

**成都福誉科技有限公司不承担由于使用本手册或本产品不当，所造成直接的、间接的、附带的或相应产品的损失或责任**

**成都福誉科技有限公司具有本产品及其软件的专利权、版权和其它知识产权。未经授权，不得直接或间接的复制、制造、加工、使用本产品及其相关部分。**

|  |  |
| --- | --- |
| **C:\Users\Administrator\AppData\Local\Temp\0d0a6f46076592e0ff9ca24f81582ff1.gif**  **注意** | **运动中的机器有危险！使用者有责任在机器中设计有效的出错处理和安全保护机制，成都福誉科技有限公司没有义务或责任对此造成的附带的或相应产生的损失责任** |

**联系我们**

**成都福誉科技有限公司**

**地址：成都市双流县西航港大道中4段1455号**

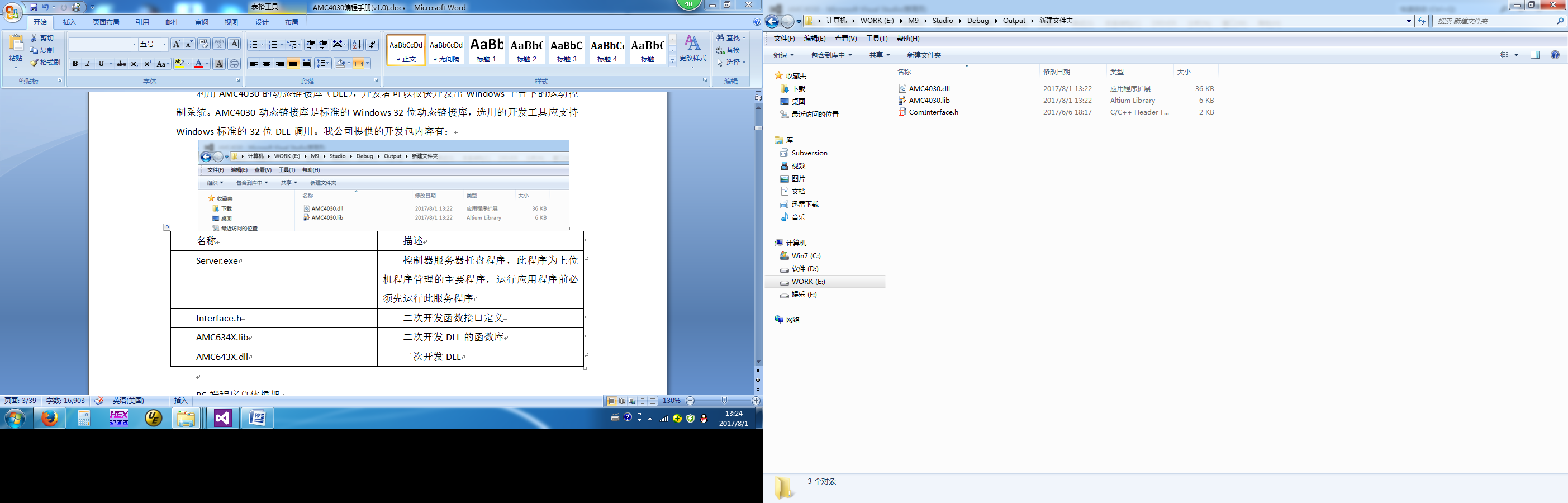
**电话：028-65058998**

**传真：028-65058997**

**网址：www.fuyuautomation.com**

# 1 概述

利用AMC4030的动态链接库（DLL），开发者可以很快开发出Windows平台下的运动控制系统。AMC4030动态链接库是标准的Windows 32位动态链接库，选用的开发工具应支持Windows标准的32位DLL调用。我公司提供的开发包内容有：



|  |  |
| --- | --- |
| 名称 | 描述 |
| ComInterface.h | 二次开发函数接口定义 |
| AMC4030.lib | 二次开发DLL的函数库 |
| AMC4030.dll | 二次开发DLL |

PC端程序总体框架

应用程序通过AMC4030.dll提供的函数接口进行二次开发，实现自己的运动控制系统。AMC4030是应用程序与AMC4030运动控制器的桥梁，通过DLL可以实现和AMC4030运动控制器的数据交互和命令传递。

AMC4030.dll

应用程序

AMC4030

运动控制器

默认约定：

* 除非有特殊说明（详见函数描述），所有函数必须在一个线程中调用。
* 所有函数接口的返回值都有统一的含义（非常重要）

|  |  |
| --- | --- |
| 返回值 | 含义 |
| 1 | 命令执行完毕，控制器已经正确接收，并返回了正确值给服务器端。 |
| -1 | 命令执行失败，命令被控制判断为非法，控制器拒绝执行此命令（有可能执行此命令的条件不具备） |
| -2 | 命令参数检查没通过，命令执行失败 |
| -4 | DLL与控制器通讯超时。 |
| -6 | 命令执行冲突，表示上一个命令还未完全执行，又有新命令来了。（表现为：一个接口调用还未返回，又开始调用新的接口了。多线程中比较普遍）。 |

## 1.1开发Visual C++控制程序

（一）说明

用户可以使用VC6.0或更高版本，来进行Windows平台下运动控制系统开发。Visual C++有两种调用动态链接库的方法，使用的文件略有不同。

（二）动态链接库函数调用方法

（1）隐式调用

隐式调用步骤如下：

（a）启动Visual C++，新建一个工程；

（b）将开发包里的动态链接库“AMC4030.dll”、“AMC4030.lib”和函数声明文件“ComInterface.h”复制到工程文件中；

（c）选择“Project”菜单下的“Settings”菜单项；

（d）切换到“Link”标签页，在“Object/library modules”栏中输入“AMC4030.lib”文件名；

（e）在应用程序中加入函数库头文件的声明文件“ComInterface.h”；

（f）在应用程序的合适位置调用接口函数。

（2）显式调用

显式调用方法需要调用Windows API函数加载和释放动态链接库。方法如下：

（a）调用Windows API函数LoadLibrary()动态加载DLL；

（b）调用Windows API函数GetProcAddress()取得将要调用的DLL中函数的指针；

（c）用函数指针调用DLL中函数完成相应功能；

（e）在程序结束时或不再使用DLL中函数时，调用Windows API函数FreeLibrary()释放动态链接库。

该方法比较繁琐。我们已经将常用的DLL函数封装成类CAMC4030DLL，并提供该类的源代码。该类含有与AMC4030DLL.h函数名及参数相同的成员函数。源代码可向福誉技术支持索要，或自己编写，源码的文件名为AMC4030DLL.cpp和AMC4030DLL.h。开发人员可将其添加进工程，在程序适当地方添加该类的对象，通过对应成员函数来调用DLL中的函数。

调用步骤如下：

（a）启动Visual C++，新建一个工程；

（b）将“Interface.h”和函数声明文件AMC4030DLL.cpp和AMC4030DLL.h复制到工程文件中；

（c）选择“Project”菜单下的“Add To Project”的子菜单“Files”项；

（d）将AMC4030DLL.cpp和AMC4030DLL.h加入到工程中；

（e）在应用程序中生成CAMC4030DLL的一个对象，调用运动函数。

以上在两种方法均为VC中调用动态链接库函数的标准方法，若要获得更具体的调用方法和帮助，请参考微软Visual Stutio开发文档MSDN或相关VC参考书籍中相应部分内容。

# 2接口开发指南

## 2.1系统初始化

### 2.1.1创建链接

int nRtn;

int nType;

COM\_API\_SetComType(2);//参数2：表示建立USB通讯链接，只能传递2.

nRtn = COM\_API\_OpenLink(3,115200); //建立连接。返回1，表示创建链接成功。

## 2.2接口函数详细说明

### 2.2.1 COM\_API\_SetComType ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_SetComType(int nType) |
| 函数功能 | **设置通讯类型，** |
| 输入参数 | Int nType --2:USB通讯。目前只支持此类通讯。 |
| 输出参数 | 无 |
| 返回值 | 其它：参见返回值统一定义 |
| 备注 |  |

### 2.2.2 Com\_API\_OpenLink()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI Com\_API\_OpenLink(int nID,int nBound) |
| 函数功能 | 建立通讯连接通道 |
| 输入参数 | Int nID --随便填，系统自动处理。  Int nBound --波特率固定值115200 |
| 输出参数 |  |
| 返回值 | 参见返回值统一定义 |
| 备注 | 在开始发送命令之前，必须调用此接口，以建立硬件通讯通道。 |

### 2.2.3 COM\_API\_GetMachineStatus ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_GetMachineStatus(unsgined char\* nStatus) |
| 函数功能 | 获取当前控制器的机器状态。 |
| 输入参数 | Unsinged char\* nStatus--当前机器的状态,将此缓冲转化为机器状态结构体。  //机器状态结构体。  typedef struct \_MACHINE\_STATUS\_  {  uint32\_t dwWorkStatus; //D0:暂停中；D1：加工；D2：点动中；D3：回零中。D4:有报警。  uint8\_t dwHomeDone; //是否正确回零。D0:X轴正确回过零；D1:Y轴正确回过零；D2:Z轴正确回过零；  uint8\_t nID; //机器逻辑编号。预留多卡共用识别标示。  uint16\_t FirmVer; //固件版本。  int32\_t nPos[3]; //轴的当前位置，此值放大了100000倍，以解决浮点传输问题。  uint32\_t RealSpeed[3]; //轴的当前速度。此值放大100000倍，以解决浮点传输问题。  uint32\_t nAlmCode; //报警编码  uint16\_t dwInputStatus; //输入口状态D0-IN1; D1-IN2; D3-IN3; D4-IN4; D5-ORG1; D6-ORG2; D7-ORG3;  uint16\_t dwOutputStatus; //输出口状态。  uint32\_t Rsv[4]; //  } MACHINE\_STATUS,\*PMACHINE\_STATUS; |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.4 COM\_API\_ReadFileData ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_ReadFileData(int nSrc,int StartAdd,int len,unsigned char\* pOutput) |
| 函数功能 | 读文件中的一部分数据，从指定位置处开始读。 |
| 输入参数 | int nSrc --- 1：参数文件；2：加工文件（未实现）；  int StartAdd ---读取的文件起始位置。  int len ---读取的长度，最长不超过500字节。  unsigned char\* pOutput ----存放数据的缓冲地址。 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 | 此接口用来实现读取运动控制器中的某个文件到PC端。读取一个文件时，必须多次调用，传递不同的读取起始地址，然后把他们组合起来，形成一个完整的文件。 |

### 2.2.5 COM\_API\_WriteFileData ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_WriteFileData(int nSrc,int StartAdd,int len,unsigned char\* pInput) |
| 函数功能 | 将一部分数据，写入到指定文件的指定位置。 |
| 输入参数 | int nSrc --- 1：参数文件；2：加工文件（未实现）；  int StartAdd ---要写入文件的起始位置。  int len ---长度，最长不超过500字节。  unsigned char\* pInput ----要写入的数据。 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 | 此接口用来实现将PC端文件写入到控制器中。写入一个文件时，必须多次调用，传递不同的起始地址，然后把他们组合起来，形成一个完整的文件。 |

### 2.2.6 COM\_API\_Jog()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_Jog(int nAxis,float fDis,float Speed) |
| 函数功能 | 轴点动。调用此函数后，运动控制器将按照设置的速度，运动指定的距离。 |
| 输入参数 | int nAxis ---轴号。0：X轴，1：Y轴。2：Z轴。  Float fDis ---运动距离  Float Speed --运动速度。 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.7 COM\_API\_StopAll()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_StopAll() |
| 函数功能 | 停止机器所有运动 |
| 输入参数 |  |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.8 COM\_API\_StopAxis()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_StopAxis(int nXAxisSet,int nYAxisSet,int nZAxisSet) |
| 函数功能 | 停止某个轴。 |
| 输入参数 | int nXAxisSet --- 1：停止X轴，0：不停止X轴  int nYAxisSet --- 1：停止Y轴，0：不停止Y轴  int nZAxisSet --- 1：停止Z轴，0：不停止Z轴 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.9 COM\_API\_Home()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_Home(int nXAxisSet,int nYAxisSet,int nZAxisSet) |
| 函数功能 | 轴回零 |
| 输入参数 | int nXAxisSet --- 1：X轴回零，0：X轴不回零  int nYAxisSet --- 1：Y轴回零，0：Y轴不回零  int nZAxisSet --- 1：Z轴回零，0：Z轴不回零 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.10 COM\_API\_SetOutputBit ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_SetOutputBit(int OutputID,int nStatus) |
| 函数功能 | 设置输出口状态 |
| 输入参数 | int OutputID --- 输出口序号1~4  int nStatus --- 输出口状态，1：低电平，0：高电平 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.11 COM\_API\_GetLastError ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_GetLastError(unsigned int\* dwErr) |
| 函数功能 | 获取上一次产生的错误代码。 |
| 输入参数 |  |
| 输出参数 | unsigned int \* dwErr ---错误码 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.12 COM\_API\_SendData ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_SendData(int nLen,unsigned char\* pData) |
| 函数功能 | 发送一帧数据或命令到控制器 |
| 输入参数 | int nLen ---发送数据的长度。  Unsinged char\* pData ---需要发送的数据。 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 | 此接口为通用数据发送接口，开发者可通过自己组织发送命令数据格式，发送到控制。命令组织格式需要和我司技术支持人员沟通核实。建议开发人员不要使用此接口。 |

### 2.2.13 COM\_API\_ReadData ()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_ReadData(int nLen,unsgined char\* pInput,unsigned char\* pOutput) |
| 函数功能 | 通用发送数据接口函数，可绕过API直接发送API接口指令到控制器 |
| 输入参数 | Int nLen --- 发送数据的长度  unsgined char\* pInput ---发送数据缓冲  unsigned char\* pOutput ---接收数据缓冲 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 | 此接口为通用数据发送接口，开发者可通过自己组织发送命令数据格式，发送到控制。命令组织格式需要和我司技术支持人员沟通核实。建议开发人员不要使用此接口。 |

### 2.2.14 COM\_API\_DowloadSystemCfg()

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_DowloadSystemCfg(char\* iniPath) |
| 函数功能 | 下载配置文件到控制器。 |
| 输入参数 | i char\* iniPath --- 配置文件的PC端存放路径。 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.15 COM\_API\_StartAutoRun(int Type, int nId, char\* FileName, int Len, int isRunOnTime)

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_StartAutoRun(int Type, int nId, char\* FileName, int Len, int isRunOnTime) |
| 函数功能 | 启动控制卡内的程序 |
| 输入参数 | Type：文件类型，程序文件为2  nId：固定数值1  FileName:长度小于等于8的字符串，文件名必须为控制卡内存在的文件  Len:文件名长度  isRunOnTime：0表示取消上电运行文件，1代表上电自动运行此次传入文件名的程序文件文件 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.16 COM\_API\_GetFileName(unsigned char\* fileinfo)

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_GetFileName(unsigned char\* fileinfo) |
| 函数功能 | 获取控制卡内的文件名 |
| 输入参数 | fileinfo：文件名结构体  struct \_FILE\_INFO\_  {  Char filename[11][10]; //文件名数组  Int file\_total; //控制卡内的文件总个数  } |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.17 COM\_API\_DeleteFile(char\* filename, int Len)

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_DeleteFile(char\* filename, int Len) |
| 函数功能 | 删除卡内的文件 |
| 输入参数 | filename：需要删除的文件名，此文件必须存在于控制卡内  Len：文件名长度 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

### 2.2.18 COM\_API\_DowloadFile(char\* szPath, int nType, int showProcess)

|  |  |
| --- | --- |
| 函数原型 | int WINAPI COM\_API\_DowloadFile(char\* szPath, int nType, int showProcess) |
| 函数功能 | 下载配置文件到控制器。 |
| 输入参数 | szPath:需要下载的文件在电脑中的路径  nType：文件类型，1为控制卡配置文件，2为加工文件，3为控制卡升级文件  showProcess：0为不显示下载进度条，1为显示下载进度条 |
| 输出参数 | 无 |
| 返回值 | 参见返回值统一定义 |
| 备注 |  |

# 3系统配置文件开发指南

系统配置文件是运动控制器能正常工作的最重要的文件，也是应用程序和控制器参数交互的纽带，应用程序根据自己工艺的需要，生成系统配置文件，然后通过文件下载接口将系统配置文件下载到运动控制器中，运动控制器读取系统配置文件中的参数，然后执行新的系统配置参数。由于系统配置文件由应用程序和控制器共同维护，所以配置文件的格式和参数名称以及参数在文件中的位置不可随意修改，否则控制器将无法正确解析出某些参数。系统配置文件的后缀名必须“ini”，文件名称由数字+字母组成，且长度不要超过8字节。在系统配置参数中有的配置参数可能无意义，或则应用程序使用不到，请不要随意修改它们。

系统配置文件格式说明

[Head] //系统文件头起始端

MachineType=4030 //控制器的型号

Version=1000 //配置文件版本

[HeadEnd] //系统文件头结束

[MachineParam] //机器参数段开始

fTimerPeriod=1.000000 //机器插补周期

fWorkPrecision=0.005000 //圆弧的拆分精度

fArcCheckPrecision=0.010000 //圆弧检测精度，用来检测3点是否能够成一个圆。

fMinLen=0.200000 //线段的最小长度，如果线段小于此值，则将被合并

fMaxFeedSpeed=48000.000000 //系统的最大进给速度（mm/min)

nAccelType=1 //系统加速度类型为S型，必须为1。(未使用）

fMaxAccelSpeed=2000.000000 //系统的拐弯加速度（mm/S2)

fAccelSpeed=3000.000000 //系统的加工加速度（mm/S2)

fJAccelSpeed=200000.000000 //系统的加加速度（mm/S3)

fFastAccelSpeed=4000.000000 //系统的空程加速度（mm/S2)

ControlFlag=1 //控制字。D0为1时，表示停止后机器自动回停靠点。

wHomePowerOn=0 //开机回原点。1为开机自动回原点。

[XAxisParam] //X轴参数段开始

nPulseFactorUp=10000.000000 //驱动器转一圈需要的脉冲数

nPulseFactorDown=31.550000 //驱动器转一圈，机器运动的距离数。

nPulseLogic=1 //驱动器脉冲的有效逻辑（高电平有效还是低电平有效）

fMaxSpeed=24000.000000 //轴的最大运动速度（mm/min）

nHomeDir=1 //回零方向

fMaxPos=1300.000000 //轴的最大行程

nEnableBacklash=0 //轴的反向间隙补偿

fBacklashLen=0.000000 //反向间隙补偿距离

fBacklashSpeed=600.000000 //反向间隙补偿的速度

fHomeSpeed=3600.000000 //轴的回零速度

fHomeCheckDis=10.000000 //无意义

fHomeZeroSpeed=600.000000 //无意义

fHomeOrgSpeed=300.000000 //无意义

fHomePosOffset=10.000000 //轴回原点后偏离原点开关的位置。

[YAxisParam]

nPulseFactorUp=10000.000000 //驱动器转一圈需要的脉冲数

nPulseFactorDown=31.550000 //驱动器转一圈，机器运动的距离数。

nPulseLogic=1 //驱动器脉冲的有效逻辑（高电平有效还是低电平有效）

fMaxSpeed=24000.000000 //轴的最大运动速度（mm/min）

nHomeDir=1 //回零方向

fMaxPos=1300.000000 //轴的最大行程

nEnableBacklash=0 //轴的反向间隙补偿

fBacklashLen=0.000000 //反向间隙补偿距离

fBacklashSpeed=600.000000 //反向间隙补偿的速度

fHomeSpeed=3600.000000 //轴的回零速度

fHomeCheckDis=10.000000 //无意义

fHomeZeroSpeed=600.000000 //无意义

fHomeOrgSpeed=300.000000 //无意义

fHomePosOffset=10.000000 //轴回原点后偏离原点开关的位置。

[ZAxisParam]

nPulseFactorUp=10000.000000 //驱动器转一圈需要的脉冲数

nPulseFactorDown=31.550000 //驱动器转一圈，机器运动的距离数。

nPulseLogic=1 //驱动器脉冲的有效逻辑（高电平有效还是低电平有效）

fMaxSpeed=24000.000000 //轴的最大运动速度（mm/min）

nHomeDir=1 //回零方向

fMaxPos=1300.000000 //轴的最大行程

nEnableBacklash=0 //轴的反向间隙补偿

fBacklashLen=0.000000 //反向间隙补偿距离

fBacklashSpeed=600.000000 //反向间隙补偿的速度

fHomeSpeed=3600.000000 //轴的回零速度

fHomeCheckDis=10.000000 //无意义

fHomeZeroSpeed=600.000000 //无意义

fHomeOrgSpeed=300.000000 //无意义

fHomePosOffset=10.000000 //轴回原点后偏离原点开关的位置。